In [1]:

*# Add the dependencies.*

**import** **pandas** **as** **pd**

**import** **os**

In [2]:

*# Files to load*

school\_data\_to\_load = "Resources/schools\_complete.csv"

student\_data\_to\_load = "Resources/students\_complete.csv"

In [3]:

*# school\_data\_to\_load = os.path.join("Resources", "schools\_complete.csv")*

*# student\_data\_to\_load = os.path.join("Resources", "students\_complete.csv")*

In [4]:

*# Read the school data file and store it in a Pandas DataFrame.*

school\_data\_df = pd.read\_csv(school\_data\_to\_load)

school\_data\_df

In [5]:

*# Read the student data file and store it in a Pandas DataFrame.*

student\_data\_df = pd.read\_csv(student\_data\_to\_load)

student\_data\_df

In [6]:

*# Determine if there are any missing values in the school data.*

school\_data\_df.count()

In [7]:

*# Determine if there are any missing values in the student data.*

student\_data\_df.count()

Out[7]:

Student ID 39170

student\_name 39170

gender 39170

grade 39170

school\_name 39170

reading\_score 39170

math\_score 39170

dtype: int64

In [8]:

*# Determine if there are any missing values in the school data.*

school\_data\_df.isnull()

In [9]:

*# Determine if there are any missing values in the student data.*

student\_data\_df.isnull().sum()

Student ID 0

student\_name 0

gender 0

grade 0

school\_name 0

reading\_score 0

math\_score 0

dtype: int64

In [10]:

*# Determine if there are not any missing values in the school data.*

school\_data\_df.notnull()

In [11]:

*# Determine if there are not any missing values in the student data.*

student\_data\_df.notnull().sum()

In [12]:

*# Determine data types for the school DataFrame.*

school\_data\_df.dtypes

Out[12]:

School ID int64

school\_name object

type object

size int64

budget int64

dtype: object

In [13]:

*# determine data types for teh student DataFrame.*

student\_data\_df.dtypes

Out[13]:

Student ID int64

student\_name object

gender object

grade object

school\_name object

reading\_score int64

math\_score int64

dtype: object

In [14]:

*# Add each prefix and suffix to remove to a list.*

prefixes\_suffixes = ["Dr. ", "Mr. ","Ms. ", "Mrs. ", "Miss ", " MD", " DDS", " DVM", " PhD"]

In [15]:

*# Iterate through the words in the "prefixes\_suffixes" list and replace them with an empty space, "".*

**for** word **in** prefixes\_suffixes:

student\_data\_df["student\_name"] = student\_data\_df["student\_name"].str.replace(word,"")

In [16]:

student\_data\_df.head()

Out[16]:

In [17]:

*# Combine the data into a single dataset.*

school\_data\_complete\_df = pd.merge(student\_data\_df, school\_data\_df, on=["school\_name", "school\_name"])

school\_data\_complete\_df.head()

Out[17]:

In [18]:

*# Get the total number of students.*

student\_count = school\_data\_complete\_df.count()

student\_count

Out[18]:

Student ID 39170

student\_name 39170

gender 39170

grade 39170

school\_name 39170

reading\_score 39170

math\_score 39170

School ID 39170

type 39170

size 39170

budget 39170

dtype: int64

In [19]:

*# Calculate the total number of schools*

school\_count = school\_data\_complete\_df["school\_name"].unique()

len(school\_count)

Out[19]:

15

In [20]:

*# Calculate the total number of schools.*

school\_count = school\_data\_df["school\_name"].count()

school\_count

Out[20]:

15

In [21]:

*# Get the total number of students*

student\_count = school\_data\_complete\_df["Student ID"].count()

student\_count

Out[21]:

39170

In [22]:

*# Calculate the total budget.*

total\_budget = school\_data\_df["budget"].sum()

total\_budget

Out[22]:

24649428

In [23]:

*# Calculate the average reading score.*

average\_reading\_score = school\_data\_complete\_df["reading\_score"].mean()

average\_reading\_score

Out[23]:

81.87784018381414

In [24]:

*# Calculate the average math score.*

average\_math\_score = school\_data\_complete\_df["math\_score"].mean()

average\_math\_score

Out[24]:

78.98537145774827

In [25]:

*# Get all students who are passing math in a new DataFrame.*

passing\_math = school\_data\_complete\_df[school\_data\_complete\_df["math\_score"] >= 70]

passing\_math.head()

Out[25]:

In [26]:

*# Get the all students that are passing math in a new DataFrame.*

passing\_reading = school\_data\_complete\_df[school\_data\_complete\_df["reading\_score"] >= 70]

passing\_reading.head()

Out[26]:

In [27]:

*# Calculate the number of students passing math.*

passing\_math\_count = passing\_math["student\_name"].count()

print(passing\_math\_count)

*# Calculate the number of students passing reading.*

passing\_reading\_count = passing\_reading["student\_name"].count()

print(passing\_reading\_count)

29370

33610

In [28]:

*# Calculate the percent that passed math.*

passing\_math\_percentage = passing\_math\_count / float(student\_count) \* 100

print(passing\_math\_percentage)

*# Calculate the percent that passed reading.*

passing\_reading\_percentage = passing\_reading\_count / float(student\_count) \* 100

print(passing\_reading\_percentage)

74.9808526933878

85.80546336482001

In [29]:

*# Calculate the overall passing percentage.*

overall\_passing\_percentage = (passing\_math\_percentage + passing\_reading\_percentage ) / 2

overall\_passing\_percentage

Out[29]:

80.39315802910392

In [30]:

*# Adding a list of values with keys to create a new DataFrame.*

district\_summary\_df = pd.DataFrame(

[{"Total Schools": school\_count,

"Total Students": student\_count,

"Total Budget": total\_budget,

"Average Math Score": average\_math\_score,

"Average Reading Score": average\_reading\_score,

"% Passing Math": passing\_math\_percentage,

"% Passing Reading": passing\_reading\_percentage,

"% Overall Passing": overall\_passing\_percentage}])

district\_summary\_df

Out[30]:

In [31]:

*# Format the "Total Students" to have the comma for a thousands separator.*

district\_summary\_df["Total Students"] = district\_summary\_df["Total Students"].map("**{:,}**".format)

district\_summary\_df["Total Students"]

Out[31]:

0 39,170

Name: Total Students, dtype: object

In [32]:

*# Format "Total Budget" to have the comma for a thousands separator, a decimal separator, and a "$".*

district\_summary\_df["Total Budget"] = district\_summary\_df["Total Budget"].map("$**{:,.2f}**".format)

district\_summary\_df["Total Budget"]

Out[32]:

0 $24,649,428.00

Name: Total Budget, dtype: object

In [33]:

*# Format the columns.*

district\_summary\_df["Average Math Score"] = district\_summary\_df["Average Math Score"].map("**{:.1f}**".format)

district\_summary\_df["Average Reading Score"] = district\_summary\_df["Average Reading Score"].map("**{:.1f}**".format)

district\_summary\_df["% Passing Math"] = district\_summary\_df["% Passing Math"].map("**{:.0f}**%".format)

district\_summary\_df["% Passing Reading"] = district\_summary\_df["% Passing Reading"].map("**{:.0f}**%".format)

district\_summary\_df["% Overall Passing"] = district\_summary\_df["% Overall Passing"].map("**{:.0f}**%".format)

In [34]:

district\_summary\_df

Out[34]:

In [35]:

*# Reorder the columns in the order you want them to appear.*

new\_column\_order = ["Total Schools", "Total Students", "Total Budget","Average Math Score", "Average Reading Score", "% Passing Math", "% Passing Reading", "% Overall Passing"]

*# Assign district summary df the new column order.*

district\_summary\_df = district\_summary\_df[new\_column\_order]

district\_summary\_df

Out[35]:

In [36]:

*# Determine the school type.*

per\_school\_types = school\_data\_df.set\_index(["school\_name"])["type"]

per\_school\_types

Out[36]:

school\_name

Huang High School District

Figueroa High School District

Shelton High School Charter

Hernandez High School District

Griffin High School Charter

Wilson High School Charter

Cabrera High School Charter

Bailey High School District

Holden High School Charter

Pena High School Charter

Wright High School Charter

Rodriguez High School District

Johnson High School District

Ford High School District

Thomas High School Charter

Name: type, dtype: object

In [37]:

*# Add the per\_school\_types into a DataFrame for testing.*

df = pd.DataFrame(per\_school\_types)

df

Out[37]:

In [38]:

*# Calculate the total student count.*

per\_school\_counts = school\_data\_df["size"]

per\_school\_counts

Out[38]:

0 2917

1 2949

2 1761

3 4635

4 1468

5 2283

6 1858

7 4976

8 427

9 962

10 1800

11 3999

12 4761

13 2739

14 1635

Name: size, dtype: int64

In [39]:

*# Calculate the total student count.*

per\_school\_counts = school\_data\_df.set\_index(["school\_name"])["size"]

per\_school\_counts

Out[39]:

school\_name

Huang High School 2917

Figueroa High School 2949

Shelton High School 1761

Hernandez High School 4635

Griffin High School 1468

Wilson High School 2283

Cabrera High School 1858

Bailey High School 4976

Holden High School 427

Pena High School 962

Wright High School 1800

Rodriguez High School 3999

Johnson High School 4761

Ford High School 2739

Thomas High School 1635

Name: size, dtype: int64

In [40]:

*# Calculate the total student count.*

per\_school\_counts = school\_data\_complete\_df["school\_name"].value\_counts()

per\_school\_counts

Out[40]:

Bailey High School 4976

Johnson High School 4761

Hernandez High School 4635

Rodriguez High School 3999

Figueroa High School 2949

Huang High School 2917

Ford High School 2739

Wilson High School 2283

Cabrera High School 1858

Wright High School 1800

Shelton High School 1761

Thomas High School 1635

Griffin High School 1468

Pena High School 962

Holden High School 427

Name: school\_name, dtype: int64

In [41]:

*# Calculate the total school budget.*

per\_school\_budget = school\_data\_df.set\_index(["school\_name"])["budget"]

per\_school\_budget

Out[41]:

school\_name

Huang High School 1910635

Figueroa High School 1884411

Shelton High School 1056600

Hernandez High School 3022020

Griffin High School 917500

Wilson High School 1319574

Cabrera High School 1081356

Bailey High School 3124928

Holden High School 248087

Pena High School 585858

Wright High School 1049400

Rodriguez High School 2547363

Johnson High School 3094650

Ford High School 1763916

Thomas High School 1043130

Name: budget, dtype: int64

In [42]:

*# Calculate the per capita spending.*

per\_school\_capita = per\_school\_budget / per\_school\_counts

per\_school\_capita

Out[42]:

Bailey High School 628.0

Cabrera High School 582.0

Figueroa High School 639.0

Ford High School 644.0

Griffin High School 625.0

Hernandez High School 652.0

Holden High School 581.0

Huang High School 655.0

Johnson High School 650.0

Pena High School 609.0

Rodriguez High School 637.0

Shelton High School 600.0

Thomas High School 638.0

Wilson High School 578.0

Wright High School 583.0

dtype: float64

In [43]:

*# Calculate the math scores.*

student\_school\_name = student\_data\_df.set\_index(["school\_name"])["math\_score"]

In [44]:

*# Calculate the average math scores.*

per\_school\_averages = school\_data\_complete\_df.groupby(["school\_name"]).mean()

per\_school\_averages

Out[44]:

In [45]:

*# Calculate the average test scores.*

per\_school\_math = school\_data\_complete\_df.groupby(["school\_name"]).mean()["math\_score"]

per\_school\_reading = school\_data\_complete\_df.groupby(["school\_name"]).mean()["reading\_score"]

In [46]:

per\_school\_math

Out[46]:

school\_name

Bailey High School 77.048432

Cabrera High School 83.061895

Figueroa High School 76.711767

Ford High School 77.102592

Griffin High School 83.351499

Hernandez High School 77.289752

Holden High School 83.803279

Huang High School 76.629414

Johnson High School 77.072464

Pena High School 83.839917

Rodriguez High School 76.842711

Shelton High School 83.359455

Thomas High School 83.418349

Wilson High School 83.274201

Wright High School 83.682222

Name: math\_score, dtype: float64

In [47]:

*# Calculate the passing scores by creating a filtered DataFrame.*

per\_school\_passing\_math = school\_data\_complete\_df[(school\_data\_complete\_df["math\_score"] >= 70)]

per\_school\_passing\_reading = school\_data\_complete\_df[(school\_data\_complete\_df["reading\_score"] >= 70)]

In [48]:

*# Calculate the number of students passing math and passing reading by school.*

per\_school\_passing\_math = per\_school\_passing\_math.groupby(["school\_name"]).count()["student\_name"]

per\_school\_passing\_reading = per\_school\_passing\_reading.groupby(["school\_name"]).count()["student\_name"]

In [49]:

*# Calculate the percentage of passing math and reading scores per school.*

per\_school\_passing\_math = per\_school\_passing\_math / per\_school\_counts \* 100

per\_school\_passing\_reading = per\_school\_passing\_reading / per\_school\_counts \* 100

In [50]:

*# Calculate the overall passing percentage.*

per\_overall\_passing\_percentage = (per\_school\_passing\_math + per\_school\_passing\_reading ) / 2

In [51]:

*# Adding a list of values with keys to create a new DataFrame.*

per\_school\_summary\_df = pd.DataFrame({

"School Type": per\_school\_types,

"Total Students": per\_school\_counts,

"Total School Budget": per\_school\_budget,

"Per Student Budget": per\_school\_capita,

"Average Math Score": per\_school\_math,

"Average Reading Score": per\_school\_reading,

"% Passing Math": per\_school\_passing\_math,

"% Passing Reading": per\_school\_passing\_reading,

"% Overall Passing": per\_overall\_passing\_percentage})

per\_school\_summary\_df.head()

Out[51]:

In [52]:

*# Format the Total School Budget and the Per Student Budget columns.*

per\_school\_summary\_df["Total School Budget"] = per\_school\_summary\_df["Total School Budget"].map("$**{:,.2f}**".format)

per\_school\_summary\_df["Per Student Budget"] = per\_school\_summary\_df["Per Student Budget"].map("$**{:,.2f}**".format)

*# Display the data frame*

per\_school\_summary\_df.head()

Out[52]:

In [53]:

*# Reorder the columns in the order you want them to appear.*

new\_column\_order = ["School Type", "Total Students", "Total School Budget", "Per Student Budget", "Average Math Score", "Average Reading Score", "% Passing Math", "% Passing Reading", "% Overall Passing"]

*# Assign district summary df the new column order.*

per\_school\_summary\_df = per\_school\_summary\_df[new\_column\_order]

per\_school\_summary\_df.head()

Out[53]:

In [54]:

*# Sort and show top five schools.*

top\_schools = per\_school\_summary\_df.sort\_values(["% Overall Passing"], ascending=**False**)

top\_schools.head()

Out[54]:

In [55]:

*# Sort and show top five schools.*

bottom\_schools = per\_school\_summary\_df.sort\_values(["% Overall Passing"], ascending=**True**)

bottom\_schools.head()

Out[55]:

In [56]:

*# Create a grade level DataFrames.*

ninth\_graders = school\_data\_complete\_df[(school\_data\_complete\_df["grade"] == "9th")]

tenth\_graders = school\_data\_complete\_df[(school\_data\_complete\_df["grade"] == "10th")]

eleventh\_graders = school\_data\_complete\_df[(school\_data\_complete\_df["grade"] == "11th")]

twelfth\_graders = school\_data\_complete\_df[(school\_data\_complete\_df["grade"] == "12th")]

In [57]:

*# Group each school Series by the school name for the average math score.*

ninth\_grade\_math\_scores = ninth\_graders.groupby(["school\_name"]).mean()["math\_score"]

tenth\_grade\_math\_scores = tenth\_graders.groupby(["school\_name"]).mean()["math\_score"]

eleventh\_grade\_math\_scores = eleventh\_graders.groupby(["school\_name"]).mean()["math\_score"]

twelfth\_grade\_math\_scores = twelfth\_graders.groupby(["school\_name"]).mean()["math\_score"]

In [58]:

*# Group each school Series by the school name for the average reading score.*

ninth\_grade\_reading\_scores = ninth\_graders.groupby(["school\_name"]).mean()["reading\_score"]

tenth\_grade\_reading\_scores = tenth\_graders.groupby(["school\_name"]).mean()["reading\_score"]

eleventh\_grade\_reading\_scores = eleventh\_graders.groupby(["school\_name"]).mean()["reading\_score"]

twelfth\_grade\_reading\_scores = twelfth\_graders.groupby(["school\_name"]).mean()["reading\_score"]

In [59]:

twelfth\_grade\_reading\_scores

Out[59]:

school\_name

Bailey High School 80.912451

Cabrera High School 84.287958

Figueroa High School 81.384863

Ford High School 80.662338

Griffin High School 84.013699

Hernandez High School 80.857143

Holden High School 84.698795

Huang High School 80.305983

Johnson High School 81.227564

Pena High School 84.591160

Rodriguez High School 80.376426

Shelton High School 82.781671

Thomas High School 83.831361

Wilson High School 84.317673

Wright High School 84.073171

Name: reading\_score, dtype: float64

In [60]:

*# Combine each Series for average math scores by school into single DataFrame.*

math\_scores\_by\_grade = pd.DataFrame({

"9th": ninth\_grade\_math\_scores,

"10th": tenth\_grade\_math\_scores,

"11th": eleventh\_grade\_math\_scores,

"12th": twelfth\_grade\_math\_scores})

math\_scores\_by\_grade.head()

Out[60]:

In [61]:

*# Combine each Series for average reading scores by school into single DataFrame.*

reading\_scores\_by\_grade = pd.DataFrame({

"9th": ninth\_grade\_reading\_scores,

"10th": tenth\_grade\_reading\_scores,

"11th": eleventh\_grade\_reading\_scores,

"12th": twelfth\_grade\_reading\_scores})

reading\_scores\_by\_grade.head()

Out[61]:

In [62]:

*# Format each grade column.*

math\_scores\_by\_grade["9th"] = math\_scores\_by\_grade["9th"].map("**{:.1f}**".format)

math\_scores\_by\_grade["10th"] = math\_scores\_by\_grade["10th"].map("**{:.1f}**".format)

math\_scores\_by\_grade["11th"] = math\_scores\_by\_grade["11th"].map("**{:.1f}**".format)

math\_scores\_by\_grade["12th"] = math\_scores\_by\_grade["12th"].map("**{:.1f}**".format)

*# Make sure the columns are in the correct order.*

math\_scores\_by\_grade = math\_scores\_by\_grade[

["9th", "10th", "11th", "12th"]]

*# Remove the index name.*

math\_scores\_by\_grade.index.name = **None**

*# Display the DataFrame.*

math\_scores\_by\_grade.head()

Out[62]:

In [63]:

*# Format each grade column.*

reading\_scores\_by\_grade["9th"] = reading\_scores\_by\_grade["9th"].map("**{:,.1f}**".format)

reading\_scores\_by\_grade["10th"] = reading\_scores\_by\_grade["10th"].map("**{:,.1f}**".format)

reading\_scores\_by\_grade["11th"] = reading\_scores\_by\_grade["11th"].map("**{:,.1f}**".format)

reading\_scores\_by\_grade["12th"] = reading\_scores\_by\_grade["12th"].map("**{:,.1f}**".format)

*# Make sure the columns are in the correct order.*

reading\_scores\_by\_grade = reading\_scores\_by\_grade[

["9th", "10th", "11th", "12th"]]

*# Remove the index name.*

reading\_scores\_by\_grade.index.name = **None**

*# Display the data frame.*

reading\_scores\_by\_grade.head()

Out[63]:

In [64]:

*# Get the descriptive statistics for the per\_school\_capita.*

per\_school\_capita.describe()

Out[64]:

count 15.000000

mean 620.066667

std 28.544368

min 578.000000

25% 591.500000

50% 628.000000

75% 641.500000

max 655.000000

dtype: float64

In [65]:

*# Cut the per\_school\_capita into the spending ranges.*

spending\_bins = [0, 585, 615, 645, 675]

pd.cut(per\_school\_capita, spending\_bins)

Out[65]:

Bailey High School (615, 645]

Cabrera High School (0, 585]

Figueroa High School (615, 645]

Ford High School (615, 645]

Griffin High School (615, 645]

Hernandez High School (645, 675]

Holden High School (0, 585]

Huang High School (645, 675]

Johnson High School (645, 675]

Pena High School (585, 615]

Rodriguez High School (615, 645]

Shelton High School (585, 615]

Thomas High School (615, 645]

Wilson High School (0, 585]

Wright High School (0, 585]

dtype: category

Categories (4, interval[int64]): [(0, 585] < (585, 615] < (615, 645] < (645, 675]]

In [66]:

*# Cut the per\_school\_capita into the spending ranges.*

spending\_bins = [585, 615, 645, 675]

pd.cut(per\_school\_capita, spending\_bins)

Out[66]:

Bailey High School (615.0, 645.0]

Cabrera High School NaN

Figueroa High School (615.0, 645.0]

Ford High School (615.0, 645.0]

Griffin High School (615.0, 645.0]

Hernandez High School (645.0, 675.0]

Holden High School NaN

Huang High School (645.0, 675.0]

Johnson High School (645.0, 675.0]

Pena High School (585.0, 615.0]

Rodriguez High School (615.0, 645.0]

Shelton High School (585.0, 615.0]

Thomas High School (615.0, 645.0]

Wilson High School NaN

Wright High School NaN

dtype: category

Categories (3, interval[int64]): [(585, 615] < (615, 645] < (645, 675]]

In [67]:

*# Cut the per\_school\_capita into the spending ranges.*

spending\_bins = [0, 585, 615, 645, 675]

per\_school\_capita.groupby(pd.cut(per\_school\_capita, spending\_bins)).count()

Out[67]:

(0, 585] 4

(585, 615] 2

(615, 645] 6

(645, 675] 3

dtype: int64

In [68]:

*# Cut the per\_school\_capita into the spending ranges.*

spending\_bins = [0, 585, 630, 645, 675]

per\_school\_capita.groupby(pd.cut(per\_school\_capita, spending\_bins)).count()

Out[68]:

(0, 585] 4

(585, 630] 4

(630, 645] 4

(645, 675] 3

dtype: int64

In [69]:

*# Establish the spending bins and group names.*

spending\_bins = [0, 585, 630, 645, 675]

group\_names = ["<$584", "$585-629", "$630-644", "$645-675"]

In [70]:

*# Categorize spending based on the bins.*

per\_school\_summary\_df["Spending Ranges (Per Student)"] = pd.cut(per\_school\_capita, spending\_bins, labels=group\_names)

per\_school\_summary\_df

Out[70]:

In [71]:

*# Calculate averages for the desired columns.*

spending\_math\_scores = per\_school\_summary\_df.groupby(["Spending Ranges (Per Student)"]).mean()["Average Math Score"]

spending\_reading\_scores = per\_school\_summary\_df.groupby(["Spending Ranges (Per Student)"]).mean()["Average Reading Score"]

spending\_passing\_math = per\_school\_summary\_df.groupby(["Spending Ranges (Per Student)"]).mean()["% Passing Math"]

spending\_passing\_reading = per\_school\_summary\_df.groupby(["Spending Ranges (Per Student)"]).mean()["% Passing Reading"]

In [72]:

*# Calculate the overall passing percentage.*

overall\_passing\_percentage = (spending\_passing\_math + spending\_passing\_reading) / 2

In [73]:

overall\_passing\_percentage

Out[73]:

Spending Ranges (Per Student)

<$584 95.035486

$585-629 89.925871

$630-644 78.938001

$645-675 73.649382

dtype: float64

In [74]:

*# Assemble into DataFrame.*

spending\_summary\_df = pd.DataFrame({

"Average Math Score" : spending\_math\_scores,

"Average Reading Score": spending\_reading\_scores,

"% Passing Math": spending\_passing\_math,

"% Passing Reading": spending\_passing\_reading,

"% Overall Passing": overall\_passing\_percentage})

spending\_summary\_df

Out[74]:

In [75]:

*# Formatting*

spending\_summary\_df["Average Math Score"] = spending\_summary\_df["Average Math Score"].map("**{:.1f}**".format)

spending\_summary\_df["Average Reading Score"] = spending\_summary\_df["Average Reading Score"].map("**{:.1f}**".format)

spending\_summary\_df["% Passing Math"] = spending\_summary\_df["% Passing Math"].map("**{:.0f}**".format)

spending\_summary\_df["% Passing Reading"] = spending\_summary\_df["% Passing Reading"].map("**{:.0f}**".format)

spending\_summary\_df["% Overall Passing"] = spending\_summary\_df["% Overall Passing"].map("**{:.0f}**".format)

spending\_summary\_df

Out[75]:

In [76]:

*# Establish the bins.*

size\_bins = [0, 1000, 2000, 5000]

group\_names = ["Small (<1000)", "Medium (1000-2000)", "Large (2000-5000)"]

In [77]:

*# Categorize spending based on the bins.*

per\_school\_summary\_df["School Size"] = pd.cut(per\_school\_summary\_df["Total Students"], size\_bins, labels=group\_names)

per\_school\_summary\_df.head()

Out[77]:

In [78]:

*# Calculate averages for the desired columns.*

size\_math\_scores = per\_school\_summary\_df.groupby(["School Size"]).mean()["Average Math Score"]

size\_reading\_scores = per\_school\_summary\_df.groupby(["School Size"]).mean()["Average Reading Score"]

size\_passing\_math = per\_school\_summary\_df.groupby(["School Size"]).mean()["% Passing Math"]

size\_passing\_reading = per\_school\_summary\_df.groupby(["School Size"]).mean()["% Passing Reading"]

size\_overall\_passing = (size\_passing\_math + size\_passing\_reading) / 2

In [79]:

*# Assemble into DataFrame.*

size\_summary\_df = pd.DataFrame({

"Average Math Score" : size\_math\_scores,

"Average Reading Score": size\_reading\_scores,

"% Passing Math": size\_passing\_math,

"% Passing Reading": size\_passing\_reading,

"% Overall Passing": size\_overall\_passing})

size\_summary\_df

Out[79]:

In [80]:

*# Formatting.*

size\_summary\_df["Average Math Score"] = size\_summary\_df["Average Math Score"].map("**{:.1f}**".format)

size\_summary\_df["Average Reading Score"] = size\_summary\_df["Average Reading Score"].map("**{:.1f}**".format)

size\_summary\_df["% Passing Math"] = size\_summary\_df["% Passing Math"].map("**{:.1f}**".format)

size\_summary\_df["% Passing Reading"] = size\_summary\_df["% Passing Reading"].map("**{:.1f}**".format)

size\_summary\_df["% Overall Passing"] = size\_summary\_df["% Overall Passing"].map("**{:.1f}**".format)

size\_summary\_df

Out[80]:

In [81]:

*# Calculate averages for the desired columns.*

type\_math\_scores = per\_school\_summary\_df.groupby(["School Type"]).mean()["Average Math Score"]

type\_reading\_scores = per\_school\_summary\_df.groupby(["School Type"]).mean()["Average Reading Score"]

type\_passing\_math = per\_school\_summary\_df.groupby(["School Type"]).mean()["% Passing Math"]

type\_passing\_reading = per\_school\_summary\_df.groupby(["School Type"]).mean()["% Passing Reading"]

type\_overall\_passing = (type\_passing\_math + type\_passing\_reading) / 2

In [82]:

*# Assemble into DataFrame.*

type\_summary\_df = pd.DataFrame({

"Average Math Score" : type\_math\_scores,

"Average Reading Score": type\_reading\_scores,

"% Passing Math": type\_passing\_math,

"% Passing Reading": type\_passing\_reading,

"% Overall Passing": type\_overall\_passing})

type\_summary\_df

Out[82]:

In [83]:

*# Formatting*

type\_summary\_df["Average Math Score"] = type\_summary\_df["Average Math Score"].map("**{:.1f}**".format)

type\_summary\_df["Average Reading Score"] = type\_summary\_df["Average Reading Score"].map("**{:.1f}**".format)

type\_summary\_df["% Passing Math"] = type\_summary\_df["% Passing Math"].map("**{:.1f}**".format)

type\_summary\_df["% Passing Reading"] = type\_summary\_df["% Passing Reading"].map("**{:.1f}**".format)

type\_summary\_df["% Overall Passing"] = type\_summary\_df["% Overall Passing"].map("**{:.1f}**".format)

type\_summary\_df

Out[83]:

In [94]:

**import** **numpy** **as** **np**

In [139]:

student\_data\_df.loc[(student\_data\_df['school\_name']=='Thomas High School') & (student\_data\_df['grade'] == '9th')]['reading\_score']

Out[139]:

37537 80

37538 71

37539 86

37540 73

37543 82

..

39152 98

39153 80

39157 79

39164 97

39167 73

Name: reading\_score, Length: 461, dtype: int64

In [144]:

student\_data\_df['reading\_score'] = np.where((student\_data\_df['school\_name']=='Thomas High School') & (student\_data\_df['grade'] == '9th'), np.nan, student\_data\_df['reading\_score'])

student\_data\_df['math\_score'] = np.where((student\_data\_df['school\_name']=='Thomas High School') & (student\_data\_df['grade'] == '9th'), np.nan, student\_data\_df['math\_score'])

In [145]:

student\_data\_df.loc[(student\_data\_df['school\_name']=='Thomas High School') & (student\_data\_df['grade'] == '9th')]['reading\_score']

Out[145]:

37537 NaN

37538 NaN

37539 NaN

37540 NaN

37543 NaN

..

39152 NaN

39153 NaN

39157 NaN

39164 NaN

39167 NaN

Name: reading\_score, Length: 461, dtype: float64

In [146]:

student\_data\_df.loc[(student\_data\_df['school\_name']=='Thomas High School') & (student\_data\_df['grade'] == '9th')]['math\_score']

Out[146]:

37537 NaN

37538 NaN

37539 NaN

37540 NaN

37543 NaN

..

39152 NaN

39153 NaN

39157 NaN

39164 NaN

39167 NaN

Name: math\_score, Length: 461, dtype: float64

In [147]:

student\_data\_df.loc[student\_data\_df['school\_name']=='Thomas High School']

Out[147]:

1635 rows × 7 columns

In [148]:

*# Combine the data into a single dataset.*

school\_data\_complete\_df = pd.merge(student\_data\_df, school\_data\_df, on=["school\_name", "school\_name"])

In [149]:

*# Calculate the average reading score.*

average\_reading\_score = school\_data\_complete\_df["reading\_score"].mean()

*# Calculate the average math score.*

average\_math\_score = school\_data\_complete\_df["math\_score"].mean()

*# Get all students who are passing math in a new DataFrame.*

passing\_math = school\_data\_complete\_df[school\_data\_complete\_df["math\_score"] >= 70]

*# Get the all students that are passing math in a new DataFrame.*

passing\_reading = school\_data\_complete\_df[school\_data\_complete\_df["reading\_score"] >= 70]

*# Calculate the number of students passing math.*

passing\_math\_count = passing\_math["student\_name"].count()

*# Calculate the number of students passing reading.*

passing\_reading\_count = passing\_reading["student\_name"].count()

*# Calculate the percent that passed math.*

passing\_math\_percentage = passing\_math\_count / float(student\_count) \* 100

*# Calculate the percent that passed reading.*

passing\_reading\_percentage = passing\_reading\_count / float(student\_count) \* 100

*# Calculate the overall passing percentage.*

overall\_passing\_percentage = (passing\_math\_percentage + passing\_reading\_percentage ) / 2

In [150]:

*# Adding a list of values with keys to create a new DataFrame.*

district\_summary\_df = pd.DataFrame(

[{"Total Schools": school\_count,

"Total Students": student\_count,

"Total Budget": total\_budget,

"Average Math Score": average\_math\_score,

"Average Reading Score": average\_reading\_score,

"% Passing Math": passing\_math\_percentage,

"% Passing Reading": passing\_reading\_percentage,

"% Overall Passing": overall\_passing\_percentage}])

district\_summary\_df

Out[150]:

In [151]:

*# Determine the school type.*

per\_school\_types = school\_data\_df.set\_index(["school\_name"])["type"]

*# Calculate the total student count.*

per\_school\_counts = school\_data\_df.set\_index(["school\_name"])["size"]

*# Calculate the total school budget.*

per\_school\_budget = school\_data\_df.set\_index(["school\_name"])["budget"]

*# Calculate the per capita spending.*

per\_school\_capita = per\_school\_budget / per\_school\_counts

*# Calculate the math scores.*

student\_school\_name = student\_data\_df.set\_index(["school\_name"])["math\_score"]

*# Calculate the average math scores.*

per\_school\_averages = school\_data\_complete\_df.groupby(["school\_name"]).mean()

*# Calculate the average test scores.*

per\_school\_math = school\_data\_complete\_df.groupby(["school\_name"]).mean()["math\_score"]

per\_school\_reading = school\_data\_complete\_df.groupby(["school\_name"]).mean()["reading\_score"]

*# Calculate the passing scores by creating a filtered DataFrame.*

per\_school\_passing\_math = school\_data\_complete\_df[(school\_data\_complete\_df["math\_score"] >= 70)]

per\_school\_passing\_reading = school\_data\_complete\_df[(school\_data\_complete\_df["reading\_score"] >= 70)]

In [152]:

*# Calculate the number of students passing math and passing reading by school.*

per\_school\_passing\_math = per\_school\_passing\_math.groupby(["school\_name"]).count()["student\_name"]

per\_school\_passing\_reading = per\_school\_passing\_reading.groupby(["school\_name"]).count()["student\_name"]

*# Calculate the percentage of passing math and reading scores per school.*

per\_school\_passing\_math = per\_school\_passing\_math / per\_school\_counts \* 100

per\_school\_passing\_reading = per\_school\_passing\_reading / per\_school\_counts \* 100

*# Calculate the overall passing percentage.*

per\_overall\_passing\_percentage = (per\_school\_passing\_math + per\_school\_passing\_reading ) / 2

In [155]:

*# Adding a list of values with keys to create a new DataFrame.*

per\_school\_summary\_df = pd.DataFrame({

"School Type": per\_school\_types,

"Total Students": per\_school\_counts,

"Total School Budget": per\_school\_budget,

"Per Student Budget": per\_school\_capita,

"Average Math Score": per\_school\_math,

"Average Reading Score": per\_school\_reading,

"% Passing Math": per\_school\_passing\_math,

"% Passing Reading": per\_school\_passing\_reading,

"% Overall Passing": per\_overall\_passing\_percentage})

per\_school\_summary\_df.head()

Out[155]:

In [158]:

*# Create a grade level DataFrames.*

ninth\_graders = school\_data\_complete\_df[(school\_data\_complete\_df["grade"] == "9th")]

tenth\_graders = school\_data\_complete\_df[(school\_data\_complete\_df["grade"] == "10th")]

eleventh\_graders = school\_data\_complete\_df[(school\_data\_complete\_df["grade"] == "11th")]

twelfth\_graders = school\_data\_complete\_df[(school\_data\_complete\_df["grade"] == "12th")]

*# Group each school Series by the school name for the average math score.*

ninth\_grade\_math\_scores = ninth\_graders.groupby(["school\_name"]).mean()["math\_score"]

tenth\_grade\_math\_scores = tenth\_graders.groupby(["school\_name"]).mean()["math\_score"]

eleventh\_grade\_math\_scores = eleventh\_graders.groupby(["school\_name"]).mean()["math\_score"]

twelfth\_grade\_math\_scores = twelfth\_graders.groupby(["school\_name"]).mean()["math\_score"]

*# Group each school Series by the school name for the average reading score.*

ninth\_grade\_reading\_scores = ninth\_graders.groupby(["school\_name"]).mean()["reading\_score"]

tenth\_grade\_reading\_scores = tenth\_graders.groupby(["school\_name"]).mean()["reading\_score"]

eleventh\_grade\_reading\_scores = eleventh\_graders.groupby(["school\_name"]).mean()["reading\_score"]

twelfth\_grade\_reading\_scores = twelfth\_graders.groupby(["school\_name"]).mean()["reading\_score"]

*# Combine each Series for average math scores by school into single DataFrame.*

math\_scores\_by\_grade = pd.DataFrame({

"9th": ninth\_grade\_math\_scores,

"10th": tenth\_grade\_math\_scores,

"11th": eleventh\_grade\_math\_scores,

"12th": twelfth\_grade\_math\_scores})

*# Combine each Series for average reading scores by school into single DataFrame.*

reading\_scores\_by\_grade = pd.DataFrame({

"9th": ninth\_grade\_reading\_scores,

"10th": tenth\_grade\_reading\_scores,

"11th": eleventh\_grade\_reading\_scores,

"12th": twelfth\_grade\_reading\_scores})

reading\_scores\_by\_grade

Out[158]:

In [159]:

math\_scores\_by\_grade

<https://github.com/amber-luk/Data-Online-Review/blob/master/Module_4-School_District_Analysis/PyCitySchools_Challenge.ipynb>